But Worse: Remaking Famous Games on a Budget as a Game Development Course

Nikolov, Ivan Adriyanov

Published in:
17th International Technology, Education and Development Conference

DOI (link to publication from Publisher):
10.21125/inted.2023.0839

Publication date:
2023

Document Version
Other version

BUT WORSE: REMAKING FAMOUS GAMES ON A BUDGET AS A GAME DEVELOPMENT COURSE

I. Nikolov

Architecture, Design and Media Technology, Aalborg University (DENMARK)

Abstract

As more and more higher education students become interested in pursuing careers in the game industry, it becomes vital that the teaching process aims to replicate the requirements and needs that future employers would pose. In addition to directly transitioning to larger companies, many students choose to stay as independent developers. To cover both cases, a new outlook is needed in game development courses that focus on multiple parts of the programming process – idea generation, design, implementation, testing and documentation. In addition to this, students need to be shown that continuous experimentation and building out their ideas is a normal process, even when the initial results may not be of the highest quality. To combat these potential mental blocks and create a more structured development and teaching process, a new course is created for up-and-coming students at Aalborg University, which focuses on remaking parts of widely known games in a self-imposed limited time and resource budget, which is documented, with the specific idea that the final product will be worse than the archetype, but the learnt experiences will be useful. It is demonstrated that students responded favourably to the course structure and were engaged throughout. Initial metrics from the self-reported time and resource budgeting also show that the students have gotten a good grasp on the different development parts. On average the student has taken between 15 and 20 hours of development time to create their mini-projects, with 7 different genres represented.

Keywords: Game development, Unity, prototyping, time budgeting, course work.

1 INTRODUCTION

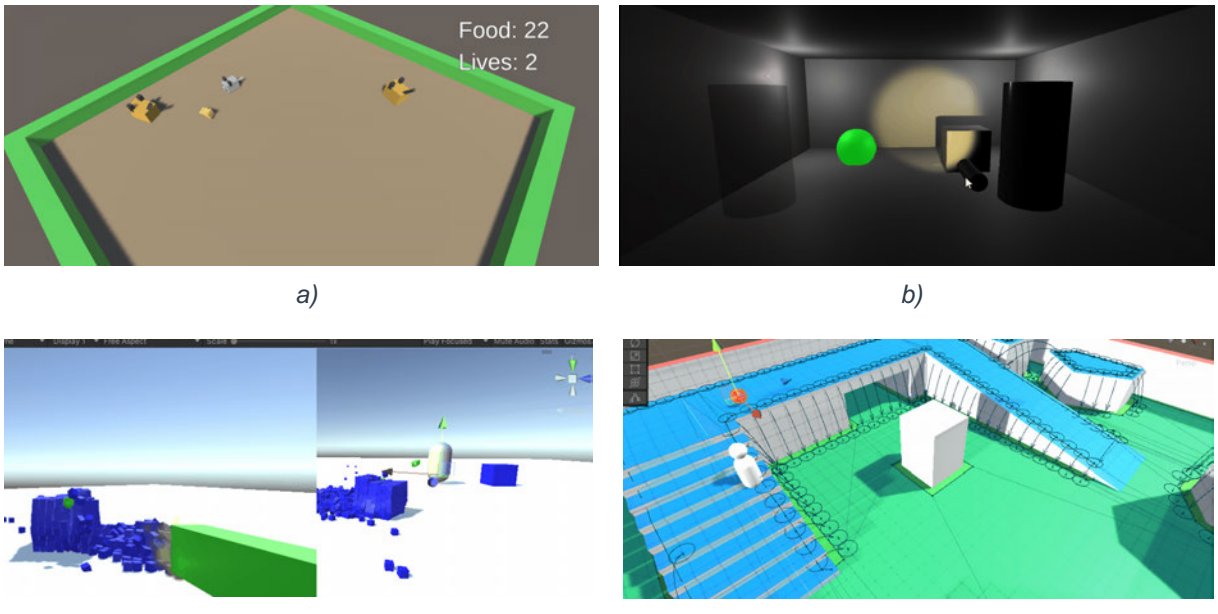

Figure 1. "But worse" example games concepts used as part of the lectures. Example mini-project game inspired by Pac-Man (Fig.1a), real-time and baked lighting using the Alan Wake flashlight mechanics (Fig. 1b), Raycasting and physics interactions framed as a shooter in the vein of Doom (Fig. 1c) and creating levels and character navigation to mimic the chase scenes in Resident Evil 2 (Fig. 1d).

The game industry is expanding at an increased rate, with revenue from free-to-play and mobile gaming providing the main growth direction [1]. As of 2022, the video game industry has surpassed all other physical and digital media, except TV [2], providing even more career possibilities for new graduates. In

addition, to commercial entertainment games, the influence of serious games [3] for primary children education [4], medicine [5][6], adult job education [7], and dissemination of important information [8][9] has also been expanding. With this growth, the number of people needing and wanting to start a career in the game industry has also increased, with only in the US their number raising above 270k in 2022 [10].

These changes have put more and more pressure on higher education institutions to remake and expand their courses, to better meet the demands of the industry. Developing such courses has been the subject of multiple research evaluations focused on testing student knowledge through the implementation of mobile games [11], game-related software solutions [12], Arduino games [13] and PC and console games [14]. Other course creation was directed towards the specific use of limited resources [15], as well as integrating industry-relevant knowledge like the Scrum working paradigm [16].

The proposed course is developed for bachelor's students in their second year of study at Aalborg University as part of the Media Technology speciality. The university uses project-based learning (PBL) and at this stage, students should transition to more individual work with an emphasis on building larger semester projects, together with mini-project around each subject. The presented course is called "Programming Interactive 3D Worlds" and focuses on building upon the student knowledge, obtained through the first year of study, by contextualizing it into developing games and interactive experiences in Unity. As part of this, the students are walked through base game development concepts like creating interactions, setting up cameras, lights and models, use cases for user interfaces, level designing, pathfinding, and physics engines, among other things using Unity. It has been observed by many that the constructivist approach to learning [17][18], where students actively participate and develop games and interactive experiences, consistently produces better learning results and longer retention of information, especially for computer science and programming [19][20].

Two main methodologies are integrated as part of the course creation – the adaptation of industry workflows, as part of the self-study part of the course and confronting and addressing the innate fears of failure in students, who start learning a subject requiring a lot of hands-on work.

To facilitate a course where students are encouraged to try out ideas and experiment without the fear of mistakes and imperfect results, the lectures are organized around the concept of presenting specific parts of game development with famous games and motivating students to remake parts of these games with a limited time and resource budget. This development direction is dubbed "But worse", where the expectations for results are lowered on purpose, with emphasis put on the learnt skills along the way. Each lecture ended with exercises which would challenge the students of creating "but worse" versions of the discussed concepts. Examples of some of the materials from the course can be seen in Fig. 1.

This "but worse" course mentality is combined with the second requirement of adapting industry-standard workflows. This is achieved by looking at the multi-stage process of game development that requires a diverse set of skills – for idea generation, defining the target audience, designing the interactions, levels, and user interface, developing and refining prototypes, testing and documenting all parts. This is done by introducing a mini-project self-study work, where students choose a famous game as the basis and try to remake it or create an inspired-by game. The students are told that they need to go through the different stages of game development and keep a time schedule showing how much time each part has taken them. The result of the mini-project should be a game prototype with code commenting, development documentation, and reference documentation describing additional learning materials and resources used, together with a project set up in a version control system like GitHub or Bitbucket. The produced prototype also needs to be a part of the "but worse" development direction. Finally, each student needed to present their project in a 5-minute format, mimicking an idea pitching session.

The course was run for 2.5 months as part of the autumn semester of 2022 and was attended by 23 students. After an overview of student feedback throughout the semester, together with an analysis of the time schedules and developed mini-projects a number of findings are reported. The students were generally positive towards the course format and expressed a sense of relief towards the idea that "excellence is not required". On average the students have worked around 20.5 hours on their mini-projects, from which 5% of the time was used for idea generation, 10% for bug-fixing and testing and 6% for code documentation, with implementation and design taking up on average 51% and 28% of the time. The self-study was reported separately and is shown that students were motivated to learn more about the inner workings of famous games to reproduce them. The course is planned to be used in further years and expanded.

The main contributions of this paper can be summed up as:

- Creating a bachelor's degree course for game and interactive experience development around the idea of reproducing famous games "but worse", while learning from the experiences and pitfalls.
- Taking the academic course closer to the industry requirements, by setting up a self-study mini-project as a game development challenge, where students need to come up with an idea based on a famous game and then develop test, document and present it as if they would be pitching a game concept to a company.

2 METHODOLOGY

The proposed course's main idea is to prepare students for working with not only game development, but interactive design and media. It is set up as groundwork that would build upon the students' knowledge in programming, user experience and developing mobile and computer applications.

The course is separated into 9 lectures, focusing on different parts of game development and design, with the final 10th lecture reserved for students pitching their game prototypes and demonstrating their work. For each of the lectures, a theme is chosen, together with an end-of-lecture exercise built around a widely known or classical game that the students can identify working with. The idea behind this is that expanding the use of widely known games from just lecture examples to part of the exercises and part of what students are trying to recreate, puts emphasis on the fact that good game design is achievable by anyone given the right tools and knowledge. It also aids understanding the exercise requirements by framing them in easily recognizable game scenarios from pop culture. Below an overview of each of the proposed lectures is given, together with the game that is used to frame it.

- Lecture 1 – Course introduction, setting up environments and main goals of the course, as well as the "but worse" idea. Going through an example game inspired by Pac-Man and Snake and identifying points of interest, what needs to be recreated, even if it is not as good as the originals
- Lecture 2 - Unity and C# crash course for getting everyone familiarized with the tools needed for the course, as well as good programming practices and prototyping concepts. Exercises framed around the game Pikmin – gathering and sending units to target enemies and objectives
- Lecture 3 – Mesh geometry and 3D transformations for game development – theory and practice. Exercises set up around the game Stellaris – recreating an interactive, moving planetary and star map
- Lecture 4 – 3D cameras, lighting, and materials in Unity theory and practical implementation for creating cohesive 3D environments. Self-study work based around recreating the light attacks and shadow enemies from Alan Wake
- Lecture 5 – Game physics theory, rigidbodies, colliders and joints in Unity for creating interactions between objects and a believable game world. As an exercise the creation of an interactive level from The Incredible Machine.
- Lecture 6 – Raycasting, physical interactions and basic AI for creating obstacles and enemies. Self-study work around duplicating the shooting and physics in the games Doom and Blood – different weapons and enemy behavior.
- Lecture 7 – Design considerations and theory behind creating a Graphical User Interface, hands-on practice with building UI in Unity. As an exercise create a skin or character selection menu in the vein of games like Fortnite and PUBG- Battlegrounds.
- Lecture 8 – Types of level design, building considerations and pathfinding. Working with Unity's built-in level design and prototyping tools and navigational system. Exercises framed around recreating a level and chase sequences like the ones from Resident Evil 2 and Resident Evil 3.
- Lecture 9 – Development optimization, workflows and best practices when creating games with Unity. How to work in collaboration and use debugging tools. Exercises based on creating an inventory menu and picking up objects, inspired by Diablo 2

The course materials and Unity projects are provided as open-source and can be accessed directly in GitHub - *https://github.com/IvanNik17/Programming-Interactive-3D-Worlds*.

The students would have between 1.5 and 2 hours to work on the exercises, while a teacher is present to help and guide them and until the following week to show their version of the required exercise.

The mini-project self-study part of the course is framed in the same "but worse" methodology, with students having the possibility to choose a favorite game or interactive experience to recreate or reimagine.

Requirements were only given for the minimal needed parts, so a cohesive experience can be achieved – at the minimum needed interactivity, a playable character, levels and obstacles or enemies, together with a GUI to interact with or display information. Students were encouraged to share ideas, help each other, and later test out and give feedback to each other. Each student had to build their own mini-project. As part of the mini-project the students should provide a prototype game, worksheets detailing the game idea, inspirations, parts of the game, as well as references to used materials and assets. In addition, timesheets need to be given with an overview of how much time has been used on different development parts like ideation, design, implementation, testing, and documentation among other things. The game and documentation need to be set up online in a version control system GitHub or Bitbucket, together with instructions and prerequisites for replication of the game. Finally, a short pitch presentation is needed to sell the idea of the mini-project. Each of these parts is selected to mimic possible challenges that students can face when looking for or starting a new job in game developer in a company or independently. To help the students an example mini-project is provided, together with documentation, code comments and a presentation. The mini-project example is also open-source and can be accessed through GitHub - *https://github.com/IvanNik17/ExampleProjects/tree/main/Pi3D-Example%20Hand-in*

3 RESULTS

The course was taken by 23 students, who handed in their mini-projects to complete it. Some examples of the created games are given in Fig.2.

a) *b)*

c) *d)*

e) *f)*

Figure 2. Examples of the different genres of mini-project games developed by the students – hack & slash based on Dynasty Warriors (Fig. 2a), flight based on Crimson Skies (Fig. 2b), horror based on Resident Evil 7 (Fig. 2c), a platformer based on Sly Cooper (Fig. 2d), exploration based on Amnesia (Fig. 2e), and stealth based Thief (Fig. 2f)

When analysing the timesheets presented by the student, we can see that on average most students have worked on their mini-projects between 15 and 20 hours, with some outliers working more than 30 and less than 11 hours, as seen in Fig. 3.

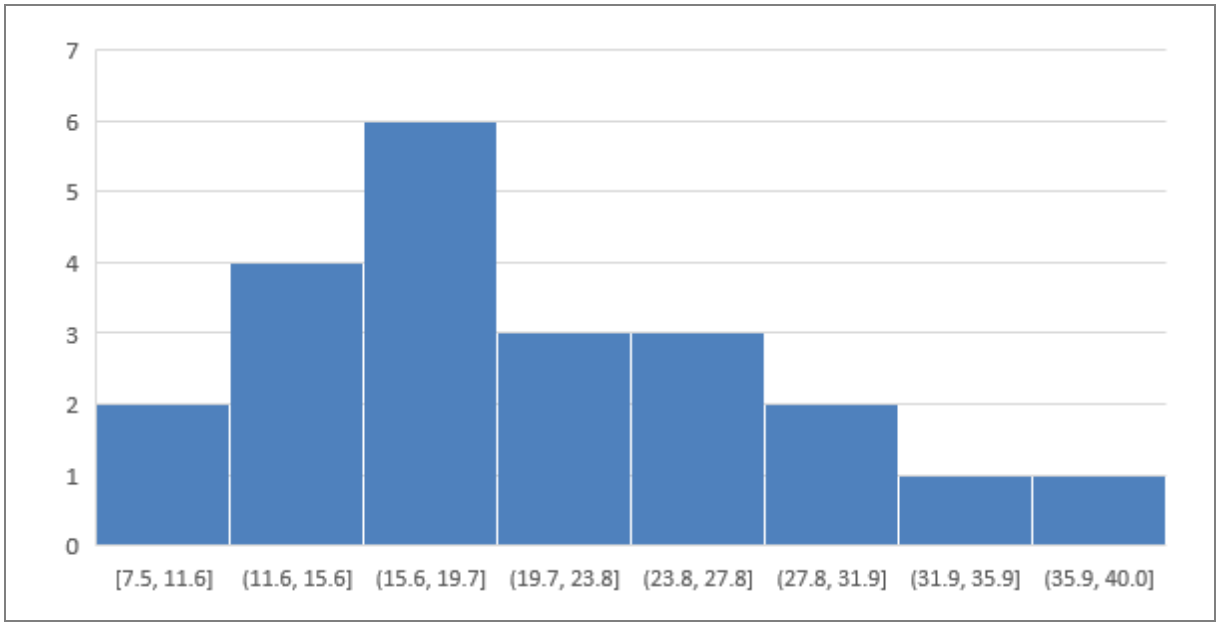

Figure 3. Total time students spent developing their mini-projects.

The main genres students have identified for their games are – first-person shooter (FPS), platformer, stealth, flight, hack & slash, horror, and exploration. Most of the games developed fall in the FPS, platformer, and stealth genres, with more details shown in Fig 4. This can be seen also as overlap with the study on using games for training computational thinking (CT) in students [21], where the main strengths of the CT score training are observed in a shooter, simulation, and platformer games. On the other hand, if we look at the average conceptualization, testing and bug fixing and code documentation times, based on the selected game genres we can see that the more seldomly selected genres like horror, flight and exploration have required much longer to do these than FPS games. This can be attributed to the relatively more straightforward development process and a larger pool of available assets, tutorials, as well as famous modern FPS games to take inspiration from. Platformer and horror games on average took the most time to test and bugfix, which can be attributed to being harder to make these kinds of games feel good and interesting to play. More in-depth information can be found in Table 1.

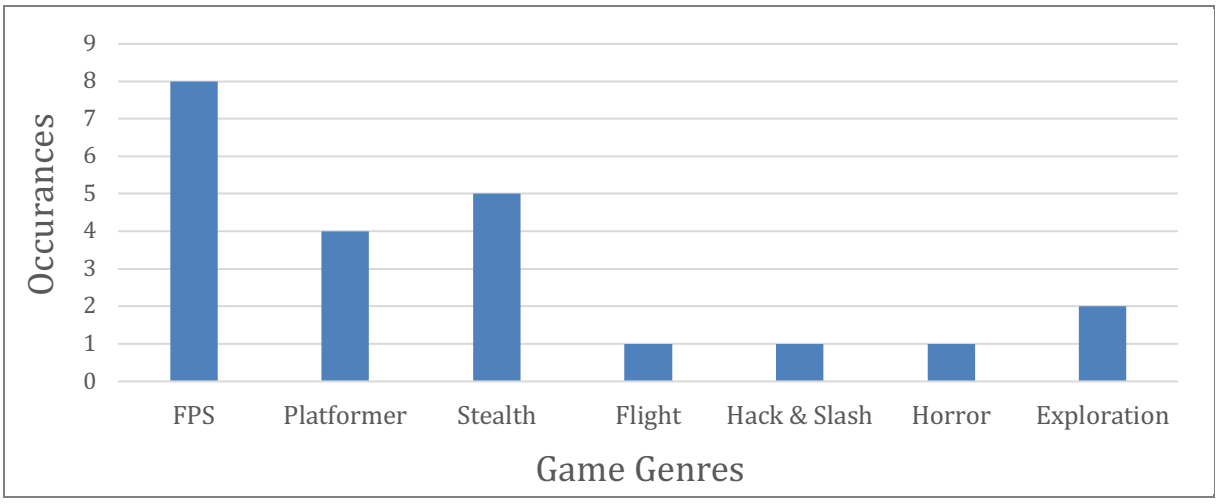

Figure 4. Number of mini-projects in each genre

Table 1. Average conceptualization, testing and documentation times per genre of the mini-projects

Genre	*Avg. Conceptualization [hours]*	*Avg. Testing/Bug fixing [hours]*	*Avg. Documentation [hours]*
FPS	1.0	1.3	1.0
Platformer	0.8	**3.6**	1.0
Stealth	1.0	1.6	1.3
Flight	**3.0**	**4.0**	1.0
Hack & Slash	0.5	1.0	1.3
Horror	1.0	**4.0**	**2.0**
Exploration	1.0	2.0	**1.6**

Throughout the course's duration, students have given feedback on the teaching, materials and mini-project as a way to judge the success of the course and to gather data for improving subsequent years. One of the main complaints about the lecture part of the course is that it was too long and left not enough time for students to get guidance for the exercises and mini-project work. As this complaint was received early in the course's life, it was addressed by shortening the lectures and interweaving the exercises through the lecture's duration, so students can practice what was presented and could get direct feedback from the lector. Another complaint that was given was that the course did not go into enough detail about developing animations and AI using unity, as well as post-processing effects. This was echoed by many students, and it was decided that the course will be changed in its next iteration to contain these two additional lectures. The main complaint to the mini-project on the other hand was the larger documentation workload, expected from the students, as compared to a pure programming-related hand-in. To address this complaint an additional short talk was organized to show students the need for proper documentation of programming work.

On the other hand, students also expressed their interest and positive reactions towards the structure of "but worse". There were multiple students sharing that they felt that this gave them a more open field to experiment and try to build game mechanics and concepts from their favourite games, without the fear of not achieving perfect results. Students were also positive towards the pitching session, where they could show their work to their piers and practice talking about it. Finally, some students shared that they have showed their mini-projects in social media and have gotten positive reactions by people recognizing the inspiration behind them.

4 CONCLUSIONS

The presented game design course was built to utilize a constructivist approach to teaching. It was designed to inspire students to experiment and develop their game ideas, without the worry that the final product will be of lesser quality. This was done by implementing the theme of "but worse" in which the results were expected to be worse than the inspiration and the resources and time given were limited with that in mind. In addition, the self-work mini-projects proposed for the course were designed to challenge the students in two ways – first by asking them to remake a widely known game of their choosing and second to create a full suite of documentation and time sheets, together with a game pitch presentation, which would mimic the required work hand-ins when working in the game industry. After the course completion, the students had positive reactions to the mini-project and the lectures and exercises, with some comments directed at possible improvements. From analyzing the students' time sheets and documentation, it can be seen that it took them between 15 and 20 hours to create the mini-projects, with most projects being shooters, platformers and stealth games.

REFERENCES

[1] Mordor Intelligence, Gaming Market – Growth, Trends, Covid-19 Impact and Forecasts (2022-2027), Accessed 21 December 2022. Retrieved from https://www.mordorintelligence.com/industry-reports/global-gaming-market

[2] A. Guttmann, Estimated media revenue worldwide in 2020, by category, 17 January 2022. Retrieved from https://www.statista.com/statistics/1132706/media-revenue-worldwide/.

[3] Zhonggen, Y., 2019. A meta-analysis of use of serious games in education over a decade. International Journal of Computer Games Technology, 2019.

[4] Tokac, U., Novak, E. and Thompson, C.G., 2019. Effects of game-based learning on students' mathematics achievement: A meta-analysis. Journal of Computer Assisted Learning, 35(3), pp.407-420.

[5] Wu, J., Loprinzi, P.D. and Ren, Z., 2019. The rehabilitative effects of virtual reality games on balance performance among children with cerebral palsy: a meta-analysis of randomized controlled trials. International journal of environmental research and public health, 16(21), p.4161.

[6] Gorbanev, I., Agudelo-Londoño, S., González, R.A., Cortes, A., Pomares, A., Delgadillo, V., Yepes, F.J. and Muñoz, Ó., 2018. A systematic review of serious games in medical education: quality of evidence and pedagogical strategy. Medical education online, 23(1), p.1438718.

[7] Larson, K., 2020. Serious games and gamification in the corporate training environment: A literature review. TechTrends, 64(2), pp.319-328.

[8] Gaspar, J.D.S., Lage, E.M., Da Silva, F.J., Mineiro, É., De Oliveira, I.J.R., Oliveira, I., De Souza, R.G., Gusmão, J.R.O., De Souza, C.F.D. and Reis, Z.S.N., 2020. A mobile serious game about the pandemic (COVID-19-Did You Know?): Design and evaluation study. JMIR serious games, 8(4), p.e25226.

[9] Nikolov, I.A. and Madsen, C.B., 2021, February. Initial Development of" Infection Defender": A Children's Educational Game for Pandemic Prevention Measurements. In Visigrapp (1: Grapp) (pp. 253-260).

[10] IbisWorld, Video Games in the US – Employment Statistics 2005-2028, Accessed 21 December 2022. Retrieved from https://www.ibisworld.com/industry-statistics/employment/video-games-united-states

[11] Kurkovsky, S., 2009. Engaging students through mobile game development. ACM SIGCSE Bulletin, 41(1), pp.44-48.

[12] Dickson, P.E., Block, J.E., Echevarria, G.N. and Keenan, K.C., 2017, June. An experience-based comparison of unity and unreal for a stand-alone 3D game development course. In Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education (pp. 70-75).

[13] Duch, P. and Jaworski, T., 2018, July. Enriching computer science programming classes with Arduino game development. In 2018 11th International Conference on Human System Interaction (HSI) (pp. 148-154). IEEE.

[14] Yanan, Q., Ning, W., Xiaoyu, W., Xiao, W. and Difang, W., 2018, May. Developing our cool console games: a teaching reform case of computer programming course. In Proceedings of ACM Turing Celebration Conference-China (pp. 127-129).

[15] Ritzhaupt, A.D., 2009. Creating a game development course with limited resources: An evaluation study. ACM Transactions on Computing Education (TOCE), 9(1), pp.1-16.

[16] Schild, J., Walter, R. and Masuch, M., 2010, June. ABC-Sprints: adapting Scrum to academic game development courses. In Proceedings of the fifth international conference on the foundations of digital games (pp. 187-194).

[17] Tekinbas, K.S., Gresalfi, M., Peppler, K. and Santo, R., 2014. Gaming the system: designing with gamestar mechanic. MIT Press.

[18] Kafai, Y.B. and Burke, Q., 2015. Constructionist gaming: Understanding the benefits of making games for learning. *Educational psychologist*, *50*(4), pp.313-334.

[19] Overmars, M., 2004. Teaching computer science through game design. Computer, 37(4), pp.81-83.

[20] Oygardslia, K., 2015, November. Students as game designers: Learning by creating game narratives in the classroom. In International Conference on Interactive Digital Storytelling (pp. 341-344). Springer, Cham.

[21] Troiano, G.M., Chen, Q., Alba, Á.V., Robles, G., Smith, G., Cassidy, M., Tucker-Raymond, E., Puttick, G. and Harteveld, C., 2020, April. Exploring how game genre in student-designed games influences computational thinking development. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (pp. 1-17).